Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de ciencias y sistemas

Organización de Lenguajes y Compiladores 2

Guía Técnica

Herberth Guillermo Obregon Espino

Carne 201314237

Guatemala, sábado, 4 de julio de 2020

Introducción

Este es un manual el cual comprende de una aplicación que sea capaz de ejecutar codigo a partir de un lenguaje formal dado, permitiendo la utilización de operaciones aritméticas de suma, resta, multiplicación y división entre números enteros y decimales.

Clases y métodos

Clases y Archivos:

Gramarica para generar el archivo ply

asignacion\_registros : ID lopcion\_asignacion tipos\_asignacion expresion

asignacion\_registros : ID tipos\_asignacion expresion

aum\_dec : ID AUMENTO

| ID DECREMENTO

casos : CASE expresion DOSPUNTOS lsentencias

l\_casos : casos

crear\_struct : STRUCT ID LLAVEIZQ lsentencias LLAVEDER

declar\_opcion : ID

declar\_opcion : ID tipos\_asignacion expresion

declar\_opcion : ID CORIZQ CORDER

declar\_opcion : ID CORIZQ expresion CORDER

declar\_opcion : ID CORIZQ CORDER tipos\_asignacion expresion

declar\_opcion : ID CORIZQ expresion CORDER tipos\_asignacion expresion

declaracion\_funciones : tipo\_variable ID PARIZQ lista\_params PARDER LLAVEIZQ lsentencias LLAVEDER

declaracion\_funciones : tipo\_variable ID PARIZQ PARDER LLAVEIZQ lsentencias LLAVEDER

declaracion\_registros : tipo\_variable lista\_ids

declaracion\_structs : STRUCT ID declar\_opcion

declaracion\_funciones : INT MAIN PARIZQ PARDER LLAVEIZQ lsentencias LLAVEDER

expresion : ID lista\_accesos

expresion : PARIZQ expresion PARDER

expresion : expresion aumento\_decremento

aumento\_decremento : AUMENTO

| DECREMENTO

expresion : expresion MAS expresion

| expresion MENOS expresion

| expresion POR expresion

| expresion DIVIDIDO expresion

| expresion MODULO expresion

| expresion AND expresion

| expresion OR expresion

| expresion MAYOR expresion

| expresion MENOR expresion

| expresion MAYORIGUAL expresion

| expresion MENORIGUAL expresion

| expresion IGUAL expresion

| expresion DIFERENTE expresion

| expresion ANDBIT expresion

| expresion ORBIT expresion

| expresion XORBIT expresion

| expresion SHIFTIZQ expresion

| expresion SHIFTDER expresion

expresion : callfuncion

expresion : PARIZQ tipo\_variable PARDER expresion

expresion : ID

lacceso : CORIZQ expresion CORDER

lacceso : PUNTO ID

lista\_accesos : lacceso

lista\_accesos : lista\_accesos lacceso

callfuncion : ID PARIZQ l\_expresion PARDER

callfuncion : ID PARIZQ PARDER

expresion : ENTERO

| DECIMAL

| CADENA

expresion : SCANF PARIZQ PARDER

expresion : expresion\_ternario

expresion : MENOS expresion %prec UMENOS

| NOT expresion

| NOTBIT expresion

expresion : LLAVEIZQ l\_expresion LLAVEDER

forinit : declaracion\_registros

| asignacion\_registros

funcion\_print : PRINT PARIZQ CADENA COMA l\_expresion PARDER

funcion\_print : PRINT PARIZQ CADENA PARDER

init : instrucciones

instrucciones : instruccion

instruccion : error PTCOMA

sentencia\_etiqueta : ID DOSPUNTOS

instruccion : declaracion\_registros PTCOMA

| asignacion\_registros PTCOMA

| declaracion\_structs PTCOMA

| declaracion\_funciones

| sentencias\_control

| funcion\_print PTCOMA

| crear\_struct PTCOMA

sentencia\_goto : GOTO ID PTCOMA

instrucciones : instrucciones instruccion

l\_casos : l\_casos casos

l\_expresion : l\_expresion COMA expresion

l\_param : tipo\_variable ID

| STRUCT ID declar\_opcion

lsent : declaracion\_registros PTCOMA

| asignacion\_registros PTCOMA

| declaracion\_structs PTCOMA

| sentencias\_control

| funcion\_print PTCOMA

| sentencia\_break PTCOMA

| sentencia\_continue PTCOMA

| sentencia\_return PTCOMA

| callfuncion PTCOMA

| sentencia\_etiqueta

| sentencia\_goto

l\_expresion : expresion

lista\_ids : lista\_ids COMA declar\_opcion

lista\_ids : declar\_opcion

lista\_params : lista\_params COMA l\_param

lista\_params : l\_param

lsentencias : lsentencias lsent

lsentencias : lsent

opcion\_asignacion : CORIZQ expresion CORDER

opcion\_asignacion : PUNTO ID

lopcion\_asignacion : lopcion\_asignacion opcion\_asignacion

lopcion\_asignacion : opcion\_asignacion

sentencia\_break : BREAK

sentencia\_continue : CONTINUE

sentencia\_default : DEFAULT DOSPUNTOS lsentencias

sentencia\_dowhile : DO LLAVEIZQ lsentencias LLAVEDER WHILE PARIZQ expresion PARDER PTCOMA

sentencia\_if : IF PARIZQ expresion PARDER LLAVEIZQ lsentencias LLAVEDER ELSE LLAVEIZQ lsentencias LLAVEDER

sentencia\_if : IF PARIZQ expresion PARDER LLAVEIZQ lsentencias LLAVEDER ELSE sentencia\_if

sentencia\_for : FOR PARIZQ forinit PTCOMA expresion PTCOMA aum\_dec PARDER LLAVEIZQ lsentencias LLAVEDER

sentencia\_if : IF PARIZQ expresion PARDER LLAVEIZQ lsentencias LLAVEDER

sentencia\_return : RETURN expresion

sentencia\_return : RETURN

sentencia\_switch : SWITCH PARIZQ expresion PARDER LLAVEIZQ l\_casos LLAVEDER

sentencia\_switch : SWITCH PARIZQ expresion PARDER LLAVEIZQ l\_casos sentencia\_default LLAVEDER

sentencia\_while : WHILE PARIZQ expresion PARDER LLAVEIZQ lsentencias LLAVEDER

sentencias\_control : sentencia\_if

| sentencia\_while

| sentencia\_dowhile

| sentencia\_switch

| sentencia\_for

expresion\_ternario : expresion TERNARIO expresion DOSPUNTOS expresion

tipo\_variable : DOUBLE

| INT

| FLOAT

| CHAR

| VOID

tipos\_asignacion : ASIG

| ASIGMAS

| ASIGMENOS

| ASIGPOR

| ASIGDIV

| ASIGMODULO

| ASIGSHIFTIZQ

| ASIGSHITFDER

| ASIGAND

| ASIGOR

| ASIGXOR

Lex1.py

from .globales2 import \*

from ply.lex import TOKEN

D = *r*'[0-9]'

L = *r*'[a-zA-Z\_]'

H = *r*'[a-fA-F0-9]'

E = *r*'[Ee][+-]?{D}+'

FS = *r*'(f|F|l|L)'

IS = *r*'(u|U|l|L)\*'

#

# Reserved keywords

#

keywords = (

'AUTO', 'BREAK', 'CASE', 'CHAR', 'CONST',

'CONTINUE', 'DEFAULT', 'DO', 'DOUBLE', 'ELSE', 'ENUM', 'EXTERN',

'FLOAT', 'FOR', 'GOTO', 'IF', 'INLINE', 'INT', 'LONG',

'REGISTER', 'OFFSETOF',

'RESTRICT', 'RETURN', 'SHORT', 'SIGNED', 'SIZEOF', 'STATIC', 'STRUCT',

'SWITCH', 'TYPEDEF', 'UNION', 'UNSIGNED', 'VOID',

'VOLATILE', 'WHILE'

)

keyword\_map = {}

for keyword in keywords:

keyword\_map[keyword.lower()] = keyword

tokens = (

'MAIN',

'DOUBLE',

'MALLOC',

'AUTO',

'BREAK',

'CASE',

'CHAR',

'CONST',

'CONTINUE',

'DEFAULT',

'DO',

'SWITCH',

'ELSE',

'ENUM',

'EXTERN',

'FLOAT',

'FOR',

'IF',

'INT',

'REGISTER',

'RETURN',

'SIZEOF',

'STRUCT',

'VOID',

'WHILE',

'PRINT',

'SCANF',

'GOTO',

'PARIZQ',

'PARDER',

'CORIZQ',

'CORDER',

'LLAVEIZQ',

'LLAVEDER',

'MAS',

'MENOS',

'POR',

'DIVIDIDO',

'MODULO',

'PUNTO',

'COMA',

'DOBLEACCESOPUNTERO',

'AND',

'OR',

'NOT',

'ANDBIT',

'ORBIT',

'NOTBIT',

'XORBIT',

'SHIFTIZQ',

'SHIFTDER',

'IGUAL',

'DIFERENTE',

'MAYORIGUAL',

'MENORIGUAL',

'AUMENTO',

'DECREMENTO',

'MAYOR',

'MENOR',

'DECIMAL',

'ENTERO',

'CADENA',

'PTCOMA',

'DOSPUNTOS',

'ID',

'ASIG',

'ASIGMAS',

'ASIGMENOS',

'ASIGPOR',

'ASIGDIV',

'ASIGMODULO',

'ASIGSHIFTIZQ',

'ASIGSHITFDER',

'ASIGAND',

'ASIGOR',

'ASIGXOR',

'TERNARIO'

)

# Tokens

t\_MAIN = *r*'main'

t\_DOUBLE = *r*'double'

t\_MALLOC = *r*'malloc'

t\_AUTO = *r*'auto'

t\_BREAK = *r*'break'

t\_CASE = *r*'case'

t\_CHAR = *r*'char'

t\_CONST = *r*'const'

t\_CONTINUE = *r*'continue'

t\_DEFAULT = *r*'default'

t\_FLOAT = *r*'float'

t\_PRINT = *r*'printf'

t\_INT = *r*'int'

t\_STRUCT = *r*'struct'

t\_IF = *r*'if'

t\_ELSE = *r*'else'

t\_WHILE = *r*'while'

t\_DO = *r*'do'

t\_FOR = *r*'for'

t\_SWITCH = *r*'switch'

t\_SCANF = *r*'scanf'

t\_GOTO = *r*'goto'

t\_RETURN = *r*'return'

t\_VOID = *r*'void'

t\_PARIZQ = *r*'\('

t\_PARDER = *r*'\)'

t\_CORIZQ = *r*'\['

t\_CORDER = *r*'\]'

t\_LLAVEIZQ = *r*'\{'

t\_LLAVEDER = *r*'\}'

t\_MAS = *r*'\+'

t\_MENOS = *r*'-'

t\_POR = *r*'\\*'

t\_DIVIDIDO = *r*'/'

t\_MODULO = *r*'%'

t\_AND = *r*'&&'

t\_OR = *r*'\|\|'

t\_NOT = *r*'!'

t\_NOTBIT = *r*'~'

t\_TERNARIO = *r*'\?'

t\_ORBIT = *r*'\|'

t\_ANDBIT = *r*'&'

t\_XORBIT = *r*'\^'

t\_SHIFTIZQ = *r*'<<'

t\_SHIFTDER = *r*'>>'

t\_IGUAL = *r*'=='

t\_DIFERENTE = *r*'!='

t\_MAYORIGUAL = *r*'>='

t\_MENORIGUAL = *r*'<='

t\_AUMENTO = *r*'\+\+'

t\_DECREMENTO = *r*'--'

t\_MAYOR = *r*'>'

t\_MENOR = *r*'<'

t\_PTCOMA = *r*';'

t\_COMA = *r*','

t\_PUNTO = *r*'\.'

t\_DOSPUNTOS = *r*':'

t\_ASIG = *r*'\='

t\_ASIGMAS = *r*'\+\='

t\_ASIGMENOS = *r*'\-\='

t\_ASIGPOR = *r*'\\*\='

t\_ASIGDIV = *r*'\/\='

t\_ASIGMODULO = *r*'\%\='

t\_ASIGSHIFTIZQ = *r*'\<\<\='

t\_ASIGSHITFDER = *r*'\>\>\='

t\_ASIGAND = *r*'\&\='

t\_ASIGOR = *r*'\|\='

t\_ASIGXOR = *r*'\^\='

t\_ignore = " \t"

*def* t\_ID(*t*):

*r*'[a-zA-Z\_][a-zA-Z\_0-9]\*'

t.type = keyword\_map.get(t.value, "ID")

return t

*def* t\_DECIMAL(*t*):

*r*'\d+\.\d+'

try:

t.value = *float*(t.value)

except *ValueError*:

print("Floaat value too large %d", t.value)

t.value = 0

return t

*def* t\_ENTERO(*t*):

*r*'\d+'

try:

t.value = *int*(t.value)

except *ValueError*:

print("Integer value too large %d", t.value)

t.value = 0

return t

*def* t\_CADENA(*t*):

*r*'(\'|\").\*?(\"|\')'

t.value = t.value[1:-1] # remuevo las comillas

return t

*def* t\_COMMENT(*t*):

*r*"//.\*\n"

t.lexer.lineno += 1

*def* t\_newline(*t*):

*r*"\n+"

t.lexer.lineno += t.value.count("\n")

*def* find\_column(*input*, *token*):

line\_start = input.rfind(input, 0, token.lexpos) + 1

return (token.lexpos - line\_start) + 1

*def* t\_error(*t*):

descripcion = "Caracter no reconocido por MinorC: " + t.value

ErrorLexico = CError(descripcion, t.lexer.lineno, find\_column(entrada, t), "lexico")

print("ErrorLexico", ErrorLexico)

lErrores.append(ErrorLexico)

t.lexer.skip(1)

IDE

“PyCharm 2019”

El software se llevó a cabo en PyCharm, ya que este IDE es un software capacitado para el uso del lenguaje python y especializado para aplicaciones, además de ser muy intuitivo con el lenguaje, proporciona recursos para la creación de interfaz gráfica donde el usuario puede crear una GUI amena y fácil para el usuario final.

Ejecución

* $ python main.py

Sistema Operativo

* MacOs 10.15.5 19F96

Version de Python

